The GNAT nonlinear model reduction method and its application to fluid dynamics problems

Kevin Carlberg[*], Julien Cortial[†], David Amsallem[‡], Matthew Zahr[§], and Charbel Farhat[¶]

Stanford University, Stanford, CA, 94305, USA

The goal of this work is to accurately evaluate large-scale, nonlinear, finite-volume-based fluid dynamics models at low computational cost. To accomplish this objective, this work employs the Gauss–Newton with approximated tensors (GNAT) nonlinear model reduction method originally presented in Ref. 1. This technique decreases the system dimension by a least-squares Petrov–Galerkin projection, and decreases computational complexity by approximating the residual and Jacobian using the "Gappy POD" method; the latter requires computing only a few rows of the approximated quantities. This work introduces an efficient implementation of the GNAT method based on a novel "sample mesh" concept tailored for the finite volume formulation. When the reduced-order model is evaluated, this approach loads into memory only the subset of the mesh needed to sample the residual and Jacobian. This minimizes required computational resources, communication overhead, and computational complexity. A post-processing step that employs only the subset of the mesh needed for computing outputs is also proposed. Results obtained for a one-dimensional shock propagation problem highlight the method's capability to decrease solution times by orders of magnitude while retaining high levels of accuracy, even in predictive scenarios. The application of GNAT to a large-scale, compressible, turbulent flow problem with over 17 million unknowns illustrates the method's favorable performance compared with other nonlinear model reduction techniques (including collocation and discrete empirical interpolation approaches), and speedups exceeding 350 with errors less than 1% are observed. Finally, results show that the sample mesh enables the GNAT model to use many fewer processors compared with the full-order simulation.

I. Introduction

Computational fluid dynamics (CFD) tools have become indispensable in many industries due to their ability to enhance the understanding of complex fluid systems, reduce design costs, and improve the reliability of engineering systems. Unfortunately, high-fidelity CFD simulations are so computationally expensive that they can take days to months to complete, even on supercomputers with thousands of cores. As a result, these simulations are impractical for time-critical applications that also demand the accuracy provided by high-fidelity CFD models. In particular, applications such as flow control, "in the field" analysis, design optimization, uncertainty quantification, and system identification require high-fidelity simulations to be completed orders of magnitude faster than is currently possible.

[*]Graduate Student, Department of Aeronautics & Astronautics, Durand Building, Room 028, 496 Lomita Mall, Stanford University, Stanford, CA 94305-3035. Tel: (650)723-8482. AIAA Member.

[†]Graduate Student, Institute for Computational and Mathematical Engineering, Durand Building, Room 028, 496 Lomita Mall, Stanford University, Stanford, CA 94305-3035. Tel: (650)723-8482.

[‡]Research Associate, Department of Aeronautics & Astronautics, Durand Building, Room 028, 496 Lomita Mall, Stanford University, Stanford, CA 94305-3035. Tel: (650)723-8482. AIAA Member.

[§]Graduate Student, Institute for Computational and Mathematical Engineering, Durand Building, Room 028, 496 Lomita Mall, Stanford University, Stanford, CA 94305-3035. Tel: (650)723-8482.

[¶]Professor, Department of Aeronautics & Astronautics, Department of Mechanical Engineering, Institute for Computational and Mathematical Engineering, Durand Building 500, Room 257, 496 Lomita Mall, Stanford University, Stanford, CA 94305-3035. Tel: (650)723-3840. AIAA Fellow.

Projection-based model reduction methods present a promising approach for realizing this goal. These methods approximate the high-fidelity model by reducing the number of equations and unknowns describing it. To do so, they employ a projection process: they compute fast "online" solutions by searching in a low-dimensional subspace that was computed *a priori* by expensive "offline" computations. Thus, the reduced-order model used for fast online computations is characterized by small-dimensional matrices that are formed by a projection process on the full-order equations.

The computational cost of assembling these small-dimensional matrices scales with the large dimension of the high-fidelity model; for this reason, projection-based model reduction approaches are efficient primarily for problems where the matrices must be constructed only once or can be assembled *a priori.* These include linear dynamical systems with time-invariance,[2] linear static systems with operators that are affine in functions of the input parameters,[3,4] and systems with at most quadratic nonlinearities.[5–7] Within these contexts, projection-based model reduction has been successfully applied to problems in aerodynamics[8–12] and aeroelasticity[13–17] to name only a few.

On the other hand, when projection is applied to linear time-varying systems, linear static systems with nonaffine parameter dependence, or general nonlinear problems, the resulting ROM is costly to evaluate. This high cost can be attributed to a well-known performance bottleneck: the full-order nonlinear function (and possibly its Jacobian) must be computed and subsequently projected for each system solve; the cost of these operations scales with the large dimension of the original system. To overcome this roadblock, several approaches have been recently proposed that reduce the computational cost of evaluating the reduced-order model.

The "empirical interpolation" method developed for linear elliptic and coercive static problems with non-affine parameter dependence, and for nonlinear elliptic and parabolic coercive problems,[18] reduces the computational cost of evaluating the nonlinear terms by interpolating their values at a few spatial locations using an empirically-derived basis. A variant of this method uses "best [interpolation] points" and a POD basis and can be found in Ref. 19. Both of these methods operate at the continuous level, assume the PDE is elliptic or parabolic, and rely on a finite element discretization. As a result, these methods have limited applicability to CFD applications that often employ a finite volume discretization and can be characterized by hyperbolic equations. To this end, researchers have developed several more general approaches that operate at the semi-discrete level—that is, at the level of the ordinary differential equation (ODE) obtained after discretizing the PDE in space. These methods can be applied in principle to computational models arising from any spatial discretization technique (e.g. finite difference, finite element, finite volume).

The trajectory piece-wise linear (TPWL) method[20] is one such approach. This technique uses a weighted combination of different linearized models that are constructed at certain points along a "training" state trajectory. However, the performance of this method relies very heavily on the chosen linearization points and weights. Furthermore, since it never queries the original high-fidelity model at non-linearization points, this method is not typically robust for problems with severe nonlinearities.

Another class of cost-reduction methods reduces the computational cost of evaluating the model by computing only *a few* entries of the nonlinear functions; we refer to this as the "function sampling" class of methods. Within this function sampling class, collocation approaches were first investigated. In Ref. 21, collocation of the nonlinear equations was carried out, followed by a least-squares solution of the resulting overdetermined nonlinear system of equations. Similarly, Ref. 22 proposed collocation followed by Galerkin projection for linear time-varying systems. In contrast to collocation methods, function reconstruction approaches (also within the function sampling class) use the sampled entries of the nonlinear functions to approximate the *entire* nonlinear functions via interpolation or least-squares regression. One such method reconstructs the nonlinear function in the least-squares sense using the same basis used to represent the state vector. This approach was developed for general nonlinear problems[23] and for nonlinear dynamical systems with explicit time integration.[24] Other approaches include semi-discrete analogs to the empirical and best points interpolation methods, which have been presented for parameterized nonlinear statics problems[25,26] and for nonlinear dynamics problems.[25]

Unfortunately, the above function reconstruction methods construct approximations in a heuristic manner, and the resulting models lack some basic mathematical properties. Furthermore, these methods have not yet been demonstrated on finite volume discretizations or aerodynamic analyses. In addition, there have been minimal demonstrations of the ability of these methods to generate good results on truly large-scale problems.[1]

Ref. 1 introduced a Gauss–Newton with approximated tensors (GNAT) model reduction method that falls within the category of function reconstruction methods. This approach mitigates the problems listed above by employing a more systematic mathematical formulation. Specifically, the method employs approximations that satisfy mathematical properties related to optimality and consistency. This method has demonstrated the ability to generate solutions with sub-5% error rates and orders of magnitude speedups on nonlinear problems ranging from structural dynamics to transmission line modeling.[27] While promising, this approach remains in its infancy. In particular, its efficient implementation in computational mechanics codes has been unresolved. Also, this method has not yet been demonstrated on finite volume problems, aerodynamics analyses, or truly large-scale problems.

This work presents several advances in the development of the GNAT method that enable it to rapidly evaluate large-scale, nonlinear, finite volume-based fluid dynamics models. First, this work introduces a novel "sample mesh" concept that leads to a very efficient computer implementation of the GNAT method for finite volume problems. In this approach, only the the subset of the original mesh that is required to sample the nonlinear function (i.e. the sample mesh) is employed for online computations. As a result, the online stage does not load the unsampled parts of the computational domain into memory. So, this technique minimizes required computational resources, communication overhead, and computational complexity. Furthermore, since this sample mesh contains all the required connectivities, boundary conditions, etc., the code treats this sample mesh as a true CFD mesh. Existing routines and data structures can therefore be used for online computations. Secondly, this work constitutes the first time any function sampling model reduction method has been applied to problems discretized by the finite volume method. Finally, this work constitutes the first time the GNAT method has been applied to a truly large-scale problem (over 10^6 unknowns).

II. Problem Formulation

II.A. Parameterized nonlinear fluid dynamics problem

Consider an ODE written in state-space form that results from the finite volume semi-discretization of a partial differential equation (PDE) governing a nonlinear fluid dynamics problem

$$\begin{aligned}\frac{dy}{dt}(t) &= F\left(y(t), t; \mu\right)\\ y(0) &= y_0(\mu),\end{aligned} \tag{1}$$

with outputs of interest

$$\begin{aligned}z &= H\left(y(t), \mu\right)\\ &= L(\mu).\end{aligned} \tag{2}$$

Here, $t \in \mathbb{R}^+$ denotes time, $y(t) \in \mathbb{R}^N$ is the (cell-averaged) conserved fluid state, $y_0 : \mathcal{D} \to \mathbb{R}^N$ is the initial condition, and $z \in \mathbb{R}^p$ denotes the outputs that are of primary interest to the analyst. The nonlinear flux is $F : \mathbb{R}^N \times \mathbb{R}^+ \times \mathcal{D} \to \mathbb{R}^N$, and both $H : \mathbb{R}^N \times \mathcal{D} \to \mathbb{R}^p$ and $L : \mathcal{D} \to \mathbb{R}^p$ provide the mapping to the outputs of interest. The vector of input parameters (e.g. shape parameters, boundary conditions, etc.) is denoted by $\mu \in \mathcal{D} \subset \mathbb{R}^d$, $\mathcal{D}$ is the input parameter domain, and $z \in \mathbb{R}^p$ represents the outputs that are of primary interest to the analyst.

II.B. Objective: time-critical prediction

Consider the following objective: given inputs $\mu^\star \in \mathcal{D}$, compute approximations to the outputs $\tilde{L}(\mu^\star) \approx L(\mu^\star)$ in a manner that is "fast" in one of the following senses:

1. The evaluation takes a sufficiently small amount of *time*. This is applicable to near-real-time prediction applications, where the objective is to compute outputs in a time smaller than some threshold value. Examples include flow control and "in the field" analysis.

[1]The exception is Ref. 26, where the authors show orders of magnitude speedup a problem with 8.5×10^6 unknowns.

2. The evaluation consumes a sufficiently small amount of *computational resources*, as measured by computational cores multiplied by time. This is relevant to many-query applications, where the objective is to evaluate the model at as many points in the input space as possible, given a fixed amount of time and processors. Examples include uncertainty quantification and design optimization.

When the number of degrees of freedom N of the high-dimensional model is sufficiently large, solving Eq. (1) with $\mu = \mu^\star$ and computing the corresponding outputs becomes a major obstacle for the stated objective.

Instead, the following two-stage "offline–online" strategy can be employed. In the offline stage, Eqs. (1)–(2) are solved for $n_{\text{train}} \geq 1$ "training" input configurations $\mathcal{D}_{\text{train}} = \{\mu^j\}_{j=1}^{n_{\text{train}}} \subset \mathcal{D}$, and results (the data) are acquired. Then, these data are used to construct a surrogate model for the original parameterized system that is capable of rapidly reproducing the behavior of the high-dimensional model at arbitrary points in $\mathcal{D}$. The online stage uses the surrogate model for time-critical predictoin.

In contrast to surrogate modeling techniques such as data-fitting methods (e.g. Kriging, response surfaces) or methods that omit problem physics (e.g. mesh coarsening), model reduction seeks to achieve real-time prediction by *approximately* solving the state equations for the online configuration $\mu^\star$ and then computing the resulting outputs. The next section provides an overview of the GNAT model reduction method developed for the purpose of time-critical prediction.

III. Overview of the GNAT model reduction method

To make this paper as self-contained as possible, this section provides an overview of the Gauss–Newton with approximated tensors (GNAT) method originally presented in Ref. 1.

III.A. Strategy and properties

Model reduction of nonlinear systems is often executed in a somewhat *ad hoc* manner; as a result, nonlinear ROMs often lack basic mathematical properties. To avoid this pitfall, the GNAT method employs a strategy that constructs approximations to meet conditions related to *optimality* and *consistency*.

In this approach, if a given model is too computationally expensive for time-critical evaluation, an approximation is introduced, resulting in another less accurate but (hopefully) more economical model. This leads to a hierarchy of models characterized by tradeoffs between accuracy and computational complexity. The approximations are constructed to generate minimal error with respect to the previous model by being both optimal and consistent.

Optimal approximation: An approximation is optimal if it leads to approximated quantities that minimize some error measure with respect to the previous model in the hierarchy. This ensures that some measure of the error monotonically decreases as the approximation spaces expand (*a priori* convergence). □

Consistent approximation: An approximation is consistent if, when implemented without snapshot compression, it introduces no additional error in the solution at the training inputs. □

As shown in Figure 1, the model hierarchy employed by the GNAT method consists of three computational models: an original model, and two increasingly "lighter" approximated versions. Each approximated model is generated by acquiring snapshots during the evaluation of the more accurate model for training inputs, then compressing the snapshots, and finally introducing the approximation that exploits the compressed snapshots.

III.B. Fully discrete computational framework

The GNAT method adopts a *fully discrete* computational framework. That is, the method introduces approximations after the PDE has been discretized in both space and time. While this may be less convenient (the reduced-order model cannot be expressed as an ODE), it will enable optimality to be achieved in the projection approximation; this will be shown in Section III.C.

To this end, consider solving Eq. (1) by an implicit time-integrator. In this case, a sequence of n_{t} (the total number of time steps) nonlinear problems arises. Each of these nonlinear problems can be written as

$$R^n(y^{n+1}; \mu) = 0 \tag{3}$$

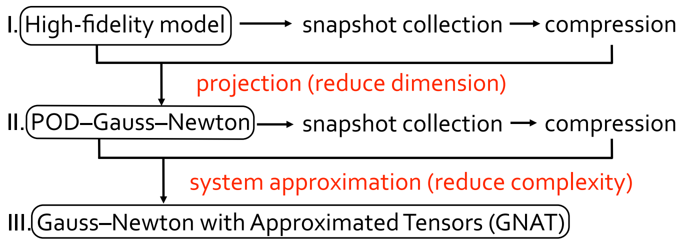

I.
High-fidelity model
snapshot collection
compression
projection (reduce dimension)
II.
POD–Gauss–Newton
snapshot collection
compression
system approximation (reduce complexity)
III.
Gauss–Newton with Approximated Tensors (GNAT)

Figure 1. Model hierarchy with approximations shown in red.

for $n = 1, \ldots, n_t$, with outputs

$$\begin{aligned} z &= G(y^0, \ldots, y^{n_t}, \mu) \\ &= L(\mu). \end{aligned} \tag{4}$$

Here, the superscript n designates the value of a variable at time step n. The operator $R^n : \mathbb{R}^N \times \mathcal{D} \rightarrow \mathbb{R}^N$ is nonlinear in at least its first argument and $G : \mathbb{R}^N \times \cdots \times \mathbb{R}^N \times \mathcal{D} \rightarrow \mathbb{R}^p$. The state variables y^n are implicitly defined by Eq. (3) given μ and R^n.

For simplicity, consider one instance (one time instance and one set of input parameters) of Eq. (3):

$$R(y) = 0. \tag{5}$$

Here, the state $y \in \mathbb{R}^N$ is implicitly defined by Eq. (5), and the mapping $R : \mathbb{R}^N \rightarrow \mathbb{R}^N$, $w \rightarrow R(w)$ is nonlinear. Eq. (5) is considered to be the equations corresponding to the full-order model (tier I in Figure 1).

III.C. Petrov–Galerkin projection

To reduce the dimension of Eq. (5), the GNAT method employs a projection process. This leads to tier II in the hierarchy. Specifically, an approximate solution $\tilde{y}$ is sought in the affine search subspace $y^{(0)} + \mathcal{Y}$ of dimension $n_y \ll N$

$$\tilde{y} = y^{(0)} + \Phi_y y_r, \tag{6}$$

where $y^{(0)} \in \mathbb{R}^N$ is an initial guess for the solution, $\Phi_y \in \mathbb{R}^{N \times n_y}$ is a basis (in matrix form) for the subspace $\mathcal{Y} \subset \mathbb{R}^N$, and $y_r \in \mathbb{R}^{n_y}$ are generalized coordinates for the state. Note that the *increment* in the state $\tilde{y} - y^{(0)}$ is sought in the subspace $\mathcal{Y}$, not the state itself; this is an important consideration when defining the basis Φ_y as will be described in Section III.C.2.

III.C.1. Optimality

In order to solve this overdetermined system, the GNAT method formulates the least squares problem

$$\underset{\bar{y} \in y^{(0)} + \mathcal{Y}}{\text{minimize}} \; \|R(\bar{y})\|_2 \tag{7}$$

and solves it using the Gauss–Newton method, which is globally convergent under certain assumptions. Note that the resulting solution is *optimal at each time step*: it minimizes the tier I residual arising at each time step over the search subspace. Also, this approach is mathematically equivalent to employing a Petrov–Galerkin projection with a test basis corresponding to the Jacobian multiplied by Φ_y (see Ref. 1).

Due to this optimality property, the Gauss–Newton (GN) tier II model has been shown to generate more accurate results than the more common Galerkin projection, which solves

$$\Phi_y^T R(y^{(0)} + \Phi_y y_r) = 0 \tag{8}$$

and is not optimal in any sense for general problems.

III.C.2. Consistency

In order for the projection to be consistent, the basis Φ_y for the search subspace should be a proper orthogonal decomposition (POD) basis[2] computed with "snapshots" collected while solving the full-order equations (tier I) at the training inputs. In Ref. 1, consistent snapshots were identified as $\{y^n - y^{n(0)}\}_{n=1}^{n_\mathrm{t}}$ where n_t is the total number of time steps for a given training input. That is, the snapshots used for POD should correspond to the solution *increment* at each time step. This is different from the approach typically taken in the literature, which is to compute a POD basis using (inconsistent) snapshots $\{y^n\}_{n=1}^{n_\mathrm{t}}$.

This work introduces yet another set of consistent snapshots: $\{y^n - y^0\}_{n=1}^{n_\mathrm{t}}$, where y^0 is the *initial condition*[3] for the dynamic simulation corresponding to time instance n.

III.D. System approximation

Solving the least-squares problem (7) by the Gauss–Newton method leads to the following iterations: for $k = 1, \ldots, K$, solve

$$p^{(k)} = \arg\min_{a \in \mathbb{R}^{n_y}} \|J^{(k)}\Phi_y a + R^{(k)}\|_2 \tag{9}$$

$$y_r^{(k+1)} = y_r^{(k)} + \alpha^{(k)} p^{(k)}, \tag{10}$$

where K is determined by the satisfaction of a convergence criterion, $y_r^{(0)} = 0$, and $R^{(k)} \equiv R(y^{(0)} + \Phi_y y_r^{(k)})$ and $J^{(k)} \equiv \nabla R\left(y^{(0)} + \Phi_y y_r^{(k)}\right)$ are the nonlinear residual and Jacobian at iteration k, respectively. The step length $\alpha^{(k)}$ is computed by executing a line search in the direction $p^{(k)}$ or is set to the canonical step length of unity. Even though the dimension of the search subspace is small, the computational cost of solving this nonlinear least-squares problem scales with the dimension N of the full-order model (tier I). The role of the system approximation (see Figure 1) is to decrease this computational cost.

III.D.1. Optimality

To do so, GNAT uses the optimal Gappy POD data reconstruction method.[28] In the context of GNAT, Gappy POD enables the quantities $R^{(k)}$ and $J^{(k)}\Phi_y$ (one- and two-dimensional tensors, respectively) to be approximated by computing only a few of their rows. Denoting by $\mathcal{I} \equiv \{\mathsf{i}_1, \mathsf{i}_2, \ldots, \mathsf{i}_{n_\mathsf{i}}\} \subset \{1, \ldots, N\}$ the set of n_i "sample indices" for which these functions are evaluated, the restriction operator $\hat{(\cdot)}$ is defined as

$$\hat{h} \equiv \begin{bmatrix} h^1_{\mathsf{i}_1} & \cdots & h^p_{\mathsf{i}_1} \\ \vdots & \ddots & \vdots \\ h^1_{\mathsf{i}_{n_\mathsf{i}}} & \cdots & h^p_{\mathsf{i}_{n_\mathsf{i}}} \end{bmatrix} = P^T h, \tag{11}$$

where $P = \begin{bmatrix} e_{\mathsf{i}_1} & \cdots & e_{\mathsf{i}_{n_\mathsf{i}}} \end{bmatrix}$ are selected columns of the identity matrix. Given these sample indices and bases $\Phi_R \in \mathbb{R}^{N \times n_\mathrm{R}}$ and $\Phi_J \in \mathbb{R}^{N \times n_\mathrm{J}}$, GNAT approximates $R^{(k)}$ and $J^{(k)}\Phi_y$ as

$$\tilde{R}^{(k)} = \Phi_R R_r^{(k)} \tag{12}$$

$$\widetilde{J^{(k)}\Phi_y} = \Phi_J J_r^{(k)}, \tag{13}$$

where $R_r^{(k)} \in \mathbb{R}^{n_R}$ and $J_r^{(k)} \in \mathbb{R}^{n_J \times n_y}$ satisfy

$$R_r^{(k)} = \arg\min_{z \in \mathbb{R}^{n_R}} \|\widehat{R^{(k)}} - \widehat{\Phi}_R z\|_2 \tag{14}$$

$$J_r^{(k)} = \arg\min_{z \in \mathbb{R}^{n_J \times n_y}} \|\widehat{J^{(k)}\Phi_y} - \widehat{\Phi}_J z\|_F \tag{15}$$

[2]A POD basis of dimension n corresponds to first n left singular vectors of the snapshot matrix, where the snapshots correspond to columns of the matrix.

[3]Note that the initial condition is the state at $t = 0$, while the initial guess $y^{n(0)}$ is taken here to be the solution at the previous time-step (i.e. y^{n-1}).

The Gappy POD reconstruction method is optimal in the sense that the error measures in Eqs. (14)–(15) monotonically decrease as the number of basis functions increases.

By introducing this approximation into Eqs. (9)–(10) and assuming $\Phi_J^T \Phi_J = I_{n_J}$, the Gauss–Newton (tier II) iterations become the GNAT (tier III) iterations

$$p^{(k)} = \arg\min_{a \in \mathbb{R}^{n_y}} \| A\widehat{J^{(k)}\Phi_y} a + B\widehat{R^{(k)}} \|_2 \tag{16}$$

$$y_r^{(k+1)} = y_r^{(k)} + \alpha^{(k)} p^{(k)}. \tag{17}$$

Here, $A = \widehat{\Phi_J}^+ \in \mathbb{R}^{n_J \times n_{\mathrm{i}}}$, $B = \Phi_J^T \Phi_R \widehat{\Phi_R}^+ \in \mathbb{R}^{n_J \times n_{\mathrm{i}}}$ and C^+ denotes the pseudo (left)-inverse of a matrix C. Note that the GNAT iterations require only:

1. computing $R^{(k)}$ and $J^{(k)}\Phi_y$ at a few sample indices, $\mathcal{I}$
2. computing the small-dimensional products $A\widehat{J^{(k)}\Phi_y}$, and $B\widehat{R^{(k)}}$
3. solving the small least-squares problem (16).

Since none of these computations scale with N, the online cost of the method will be independent of N and can therefore be very small.

III.D.2. Consistency

For the system approximation to be consistent, the bases Φ_R and Φ_J should be constructed by POD using specific snapshots. Ref. 1 provides three sufficient conditions (all three together are sufficient) on the snapshots that ensure consistency in the system approximation. These conditions lead to a hierarchy of snapshot collection procedures that trade consistency for offline cost/storage. Table 1 summarizes these procedures.

Procedure identifier	0	1	2	3
Snapshots for $R^{(k)}$	$R_{\mathrm{I}}^{(k)}$	$R_{\mathrm{II}}^{(k)}$	$R_{\mathrm{II}}^{(k)}$	$R_{\mathrm{II}}^{(k)}$
Snapshots for $J^{(k)}\Phi_y$	$R_{\mathrm{I}}^{(k)}$	$R_{\mathrm{II}}^{(k)}$	$\left[J^{(k)}\Phi_y p^{(k)}\right]_{\mathrm{II}}$	$\left[J^{(k)}\Phi_y\right]_{\mathrm{II}}$
# simulations per training input	1	2	2	2
# snapshots per Newton iteration	1	1	2	$n_y + 1$
# consistency conditions satisfied	0	1	2	3

Table 1. Snapshot collection procedures for the GNAT (tier III) model. The indicated snapshots are saved at each Newton iteration. Subscripts I and II specify the tier of the model for which the snapshots are collected.

Procedure 3, which is consistent because it satisfies all three consistency conditions, is infeasible for most problems as it requires storing $n_y + 1$ vectors at each Newton step. Procedure 2, which is not consistent but satisfies two consistency conditions, is computationally feasible as it requires saving only two vectors per Newton step. However, it employs bases $\Phi_R \neq \Phi_J$ which 1) requires three POD computations offline (Φ_y, Φ_R, and Φ_J) and 2) can make it difficult to converge iterations (16)–(17), as the least-squares problem tries to "match" quantities that lie in different subspaces.[4] Procedure 1 is more economical than Procedure 2, as it requires saving only one vector per Newton iteration and computing only two POD bases offline. Furthermore, it uses the same POD bases for the state and Jacobian, which can abet convergence of iterations (16)–(17); yet, it only satisfies one consistency condition. Procedure 0, which is similar to the approach commonly adopted in the literature,[18,19,25,26,29] satisfies none of these consistency conditions, but requires only one simulation (tier I) to be executed for each training input. For the above reasons, snapshot procedures 1 and 2 are recommended.

[4]This phenomenon has been observed in numerical experiments, although these results are not reported in this paper.

III.E. Offline–online splitting

The steps to execute the GNAT method using a "global ROM" approach as follows.

Offline

1. Execute tier I and (if using snapshot methods 1, 2, or 3 from Table 1) tier II simulations at training inputs $\mathcal{D}_{\text{train}}$. Collect snapshots of the state, residual, and Jacobian during these training simulations.
2. Compute POD bases Φ_y, Φ_R and Φ_J. To ensure uniqueness in the reconstructed gappy quantities, $n_R \geq n_y$ and $n_J \geq n_y$ are required.
3. Determine the set of n_{i} indices $\mathcal{I} \equiv \{\mathsf{i}_1, \mathsf{i}_2, \ldots, \mathsf{i}_{n_{\mathsf{i}}}\}$ for reconstructing the nonlinear functions. Section IV.B presents a method to do this for finite volume problems. To ensure uniqueness in the least-squares gappy reconstruction, the conditions $n_{\mathsf{i}} \geq n_R$ and $n_{\mathsf{i}} \geq n_J$ are required.
4. Compute the matrices $A = \widehat{\Phi_J}^{+}$ and $B = \Phi_J^T \Phi_R \widehat{\Phi_R}^{+}$ needed for online computations.
5. Generate sample mesh (see Section IV.A) and post-processing mesh (see Section IV.C).

Online

1. Evaluate the GNAT model for online inputs $\mu^\star \in \mathcal{D}$ using Algorithm 1.
2. Compute outputs of interest $G(y^0, \ldots, y^{n_{\text{t}}}, \mu^*)$ as a post-processing step. Section IV.C discusses this in the context of the proposed sample mesh concept.

Algorithm 1 Online step 1: evaluation of GNAT model

```
Input: Online matrices $A$ and $B$, initial condition $y^0$
Output: Generalized coordinates $y_r^n$, $n = 1, \ldots, n_{\text{t}}$
  Define initial condition
  for $n = 0, \ldots, n_{\text{t}} - 1$ do
    $k \leftarrow 0$
    $y^{(0)} \leftarrow y^n$, $y_r^{(0)} \leftarrow 0$
    while not converged do
      Compute $\widehat{J^{(k)}\Phi_y}$ and $\widehat{R^{(k)}}$
      Solve Eqs. (16)–(17) for $y_r^{(k+1)}$.
      $y^{(k+1)} \leftarrow y^{(k)} + \Phi_y y_r^{(k+1)}$
      $k \leftarrow k + 1$
    end while
    Write out $y_r^{n+1} = y_r^{(k)}$ for computing outputs in post-processing step
    $y^{n+1} \leftarrow y^{(k)}$
  end for
```

IV. GNAT implementation: sample mesh concept

In order to solve the GNAT iterations in Eqs. (16)–(17) as efficiently as possible, this work introduces a novel "sample mesh" approach for fluid dynamics problems discretized by the finite volume method.

In CFD codes, the most straightforward approach to solve Eqs. (16)–(17) is to load the entire computational mesh corresponding to the full-order model, update the state at the indices required to compute the nonlinear functions $J^{(k)}\Phi$ and $R^{(k)}$ at the sample indices, compute the nonlinear functions at the sample indices, and finally solve (16)–(17). Unfortunately, this approach suffers from several major drawbacks:

- Storing the full-order mesh may require many proessors. This can unnecessarily tie up computational resources, as most of the mesh is not used during online computations.
- Since the sample indices can be distributed across many processors, there may be significant communication overhead.

- There will be a non-negligible computational cost arising from searching through large quantities of memory (the uncomputed residual and Jacobian entries) in order to retrieve a small amount of data from the sample indices.

In this context, there is also a problem with adapting a straightforward greedy approach for index selection such as those suggested in Refs. [25, 29]. Namely, these approaches may be biased toward one of the state variables. Since the state variables may exhibit different scales during the simulation (e.g. the energy density may be have a larger magnitude than the mass density throughout the mesh), only indices for the large-magnitude variables might be selected for sampling (e.g. the mass density indices may never be sampled). This would lead to an unbalanced treatment of the state variables (e.g. the simulation might never query mass conservation equations).

To mitigate these problems, this work proposes a node-based "sample mesh" approach. In this approach, "sample *nodes*" are selected at which $R^{(k)}$ and $J^{(k)}\Phi_y$ are evaluated *for all state variables.* The state must be computed only at nodes that influence $R^{(k)}$ and $J^{(k)}\Phi_y$ at the sample nodes. When second-order flux reconstruction is employed, this means that the state must be computed for two layers of neighbor nodes adjacent to sample nodes. This is shown in Figure 2. Computing the residual for all state variables at a given sample node (red) requires the state to be computed at the following nodes: the sample node itself, neighbors of the sample node (blue), and finally neighbors of those neighbors (green). As a result, many nodes (depicted in black) are not needed by the GNAT method at all. In three dimensions, this effect is exaggerated—only a very small fraction of the nodes may be actually needed to execute the GNAT method.

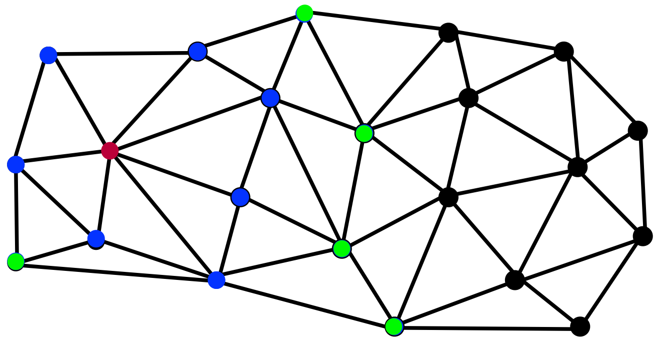


Figure 2. 2-D depiction of nodes required by the GNAT method. The residual must be evaluated for all state variables at the red "sample node." The state must be additionally evaluated at the blue and green nodes for second-order flux reconstruction. Black nodes are not needed by the method for online computations.

IV.A. Procedure

This work proposes an efficient online implementation that consists of *generating* this sample mesh during the offline stage (offline step 5 in Section III.E) . This sample mesh contains all the information characteristic of typical meshes: 1) the locations of the sample nodes and neighboring nodes, 2) connectivities (i.e. volumes), 3) boundary condition information if the sample node lies on a domain boundary, 4) "wall distance" information for certain turbulence models. As a result, it can be treated like any (disconnected) mesh: it can be partitioned, distributed across cores, etc. However, since this mesh is stripped of unneeded nodes, it is very compact and fits on a *small number of cores* relative to the original mesh. This approach is effective because:

- It minimizes computational resources. A small number cores is needed since only the required parts of the domain are loaded into memory.
- It minimizes communication overhead. Since fewer cores are used, less data must be communicated between processors.
- It minimizes computational complexity. Since only the required parts of the mesh are loaded, the algorithm does not search through large quantities of unused data to extract the needed information.
- It treats all state variables in a balanced manner.

Practically, implementing this approach is done using the following steps:

1. Choose sample nodes at which the residual will be evaluated. Section IV.B describes a node-based (not index-based) greedy method to do so. Add these nodes to the node set.
2. Add layers of neighbor nodes to the node set as appropriate (e.g. two layers if a second-order flux reconstruction is used). The state will be evaluated at these nodes.
3. Determine the connectivity between these nodes (i.e. volumes). Note that the mesh will be disconnected since some sample nodes will be isolated spatially from others.
4. Find all edges/faces contained in the sample mesh that correspond to boundary conditions, and add these to the mesh description.

In step 1, note that sample nodes can be chosen such that important parts of the domain are sensed. For example, in order to handle boundary conditions, it is important that at least one sample node lies on the boundary. If shape variables are included in the inputs, at least one sample node must be affected by the shape change in order to detect variations in the input parameter. In this way, the sample nodes can be viewed as sensors for the problem's physics.

In addition to the sample mesh just described, solving the online stage of the GNAT method (i.e. solving Eqs. (16)–(17)) requires matrices A and B and a list of the sample nodes.[5]

Once the sample mesh, online matrices, and sample nodes have been determined in the offline stage, the GNAT model is defined and can be used to make online predictions.

IV.B. Sample node selection

This section proposes a sample index selection method that, rather than choosing sample indices individually, chooses *nodes* in the mesh at which to sample the nonlinear functions. As previously discussed, this approach treats all state variables in a balanced way. The number of sample indices corresponding to each sample node is equal to the number of equations associated with each node. For example, this is often five for three-dimensional fluid problems. Thus, the sample index set $\mathcal{I}$ consists of the indices associated with degrees of freedom of the sample node set $\mathcal{N} \equiv \{\mathsf{n}_1, \mathsf{n}_2, \ldots, \mathsf{n}_{n_{\mathsf{sample}}}\}$.

The sample node selection algorithm, which is a slight variant of that proposed in Ref. 1, is provided in Algorithm 2. The method greedily chooses nodes that minimize the error in the gappy reconstruction of the nonlinear functions.

IV.C. Output computation

Oftentimes, the analyst is interested in computing outputs that cannot be directly obtained from the online evaluation of the GNAT model. In particular, computing the lift and drag requires performing calculations with parts of the mesh in contact with the surface of the immersed body. Since the sample mesh is unlikely to contain these portions of the mesh, it is not generally possible to compute such outputs online with the sample mesh alone.

Instead, a post-processing stage (online step 2 in Section III.E) can be used to compute the outputs of interest. This stage reads in the generalized POD coefficients for the state at each time step, assembles the state on a *post-processing mesh*, and then computes the outputs of interest. Algorithm 3 describes these steps.

The post-processing mesh contains only the part of the full mesh required for output computation. For the lift and drag, this corresponds to a "surface mesh" consisting of all volumes (with associated nodes, etc.) that are in contact with the immersed body. Similar to the sample mesh, the post-processing mesh minimizes computational resources, communication overhead, and computational complexity during the post-processing stage (which is online). Note that global outputs (e.g. total energy of the flow) do not lend themselves to lightweight post-processing meshes; as a result, the post-processing stage may require more computational resources in these scenarios.

[5]Recall that not all of the nodes in the sample mesh are sample nodes (e.g. Figure 2 shows that only the red node is sampled, but all colored nodes are included in the sample mesh).

Algorithm 2 Greedy algorithm for computing sample nodes

```
Input: $\Phi_R$, $\Phi_J$, number of sample nodes $n_{\mathsf{sample}}$, number of basis vectors to use $n_{\mathsf{greed}}$
Output: Sample nodes $\mathcal{N}$
 1: $\mathcal{N} = \emptyset$
 2: Determine number of greedy iterations $P$
 3: Determine number of basis vectors to use per iteration $Q$
 4: $\left[\mathsf{R}^1 \ \cdots \ \mathsf{R}^Q\right] \leftarrow \left[\phi_R^1 \ \cdots \ \phi_R^Q\right]$
 5: $\left[\mathsf{J}^1 \ \cdots \ \mathsf{J}^Q\right] \leftarrow \left[\phi_J^1 \ \cdots \ \phi_J^Q\right]$
 6: for $p = 1, \ldots, P$ do {Greedy iteration loop}
 7:     Determine number of sample nodes to add this iteration $S$
 8:     for $s = 1, \ldots, S$ do {Sample node loop}
 9:         $n \leftarrow \arg \max_{l \in \{1,\ldots,n_{\mathrm{nodes}}\} \setminus \mathcal{N}} \sum_{q=1}^{Q} \|\mathsf{R}^q\,[l]\,\|^2 + \sum_{q=1}^{Q} \|\mathsf{J}^q\,[l]\,\|^2$. Here, $\mathsf{R}^q\,[l]$ is the vector of $\mathsf{R}$ at indices associated with node $l$.
10:         $\mathcal{N} \leftarrow \mathcal{N} + n$
11:     end for
12:     for $q = 1, \ldots, Q$ do
13:         $\mathsf{R}^q \leftarrow \phi_R^{Qp+q} - \left[\phi_R^1 \ \cdots \ \phi_R^{Qp}\right] \phi_{Rr}^{Qp+q}$, with $\phi_{Rr}^{Qp+q} = \arg\min_{a \in \mathbb{R}^n} \left\| \left[\hat{\phi}_R^1 \ \cdots \ \hat{\phi}_R^{Qp}\right] a - \hat{\phi}_R^{Qp+q} \right\|_2$
14:         $\mathsf{J}^q \leftarrow \phi_J^{Qp+q} - \left[\phi_J^1 \ \cdots \ \phi_J^{Qp}\right] \phi_{Jr}^{Qp+q}$, with $\phi_{Jr}^{Qp+q} = \arg\min_{a \in \mathbb{R}^n} \left\| \left[\hat{\phi}_J^1 \ \cdots \ \hat{\phi}_J^{Qp}\right] a - \hat{\phi}_J^{Qp+q} \right\|_2$
15:     end for
16: end for
```

Algorithm 3 Online step 2: output computation

```
Input: Generalized coordinates $y_r^n$, $n = 1, \ldots, n_{\mathrm{t}}$; initial condition $y^0$ and state POD basis $\Phi_y$ in coordinates of the post-processing mesh
Output: Outputs $z$
  for $n = 1, \ldots, n_{\mathrm{t}}$ do
      $y^n = y^0 + \Phi_y y_r^n$ outputs
  end for
  Compute outputs $z = G(y^0, \ldots, y^{n_{\mathrm{t}}}, \mu^\star)$.
```

V. Applications and performance assessment

Ref. 1 demonstrated the capability of the GNAT method to generate solutions with sub-5% errors and speedups exceeding 100 on structural dynamics problems discretized by the finite element method. This section demonstrates the method's ability to generate fast, accurate solutions for problems in fluid dynamics that are discretized by the finite volume method.

In fact, this is the first time in the literature that results for a nonlinear model reduction method based on function sampling have been shown for problems modeled with a finite volume discretization.

V.A. One-dimensional inviscid Burgers' equation

To illustrate the GNAT method on a simple fluids problem, consider the one-dimensional inviscid Burgers' equation[20]

$$\frac{\partial U(x;t)}{\partial t} + 0.5\frac{\partial \left(U^2\left(x;t\right)\right)}{\partial x} = 0.02e^{bx} \tag{18}$$

with initial and boundary conditions

$$U(x,0) = 1,\ \forall x \in [0, L] \tag{19}$$

$$U(0,t) = a,\ \forall t > 0. \tag{20}$$

The length is set to $L = 100$, the inlet boundary condition is $a = \sqrt{5}$, and the coefficient in the source term is $b = 0.02$. This study employs 4001 cell centers located at $x_i = i(L/4000)$, $i = 0, \ldots, 4001$, leading to $N = 4000$ degrees of freedom. The problem is discretized using Godunov's scheme, which leads to a finite volume formulation. Note that since there is only one unknown per grid point, each sample index corresponds to a single grid point.

V.A.1. Comparison with other methods

The following procedure is employed to compare the model reduction methods. First, the full-order model (tier I) characterized by Eqs.(18)–(20) is solved and n_t (consistent) snapshots $x^i = y^n_{\mathrm{I}} - y^0$, $1 \leq i \leq n_t$ are collected. Then, the same problem (same boundary conditions, initial conditions, source term, etc.) is solved using the reduced-order models. The Gauss–Newton (II:Gauss–Newton) method uses a POD basis (computed using the above snapshots) of dimension 40; this is only 1% of the dimension of the full-order model. The TPWL method employs a POD-Galerkin projection and uses 20 linearization points determined using the trajectory curvature method,[27] as this method outperformed the trajectory distance and residual distance algorithms for this problem. The GNAT (III:GNAT(2)) method employs snapshot collection procedure 2 and parameters $n_{\mathsf{R}} = 130$, $n_{\mathsf{J}} = 40$, and $n_{\mathsf{i}} = 130$; a parametric study determined these to be optimal for this problem.

Figure 3 shows the responses computed using these models. Table 2 provides the associated speedups and errors in the time-averaged Euclidean norm of the state vector.

Method	II:Gauss–Newton	III:TPWL	III:GNAT(2)
relative error	2.66%	7.24%	2.82%
speedup	1.19	19,445	59.7

Table 2. Comparison of model reduction methods on the 1-D inviscid Burgers' equation. Relative error is the time-averaged Euclidean norm of the error in the state vector.

While II:Gauss–Newton generates an accurate solutions with sub-3% errors, its speedup was modest: only about 1.20. This illustrates the need for a system approximation to accelerate the solution computation. The III:TPWL solution exhibits severe oscillations immediately; for many fluids applications, these oscillations make the results unusable. This leads to a relative error of 7.24%; however, it generates an impressive speedup of 19,445. The III:GNAT(2) solution is much more well-behaved than the TPWL response. The

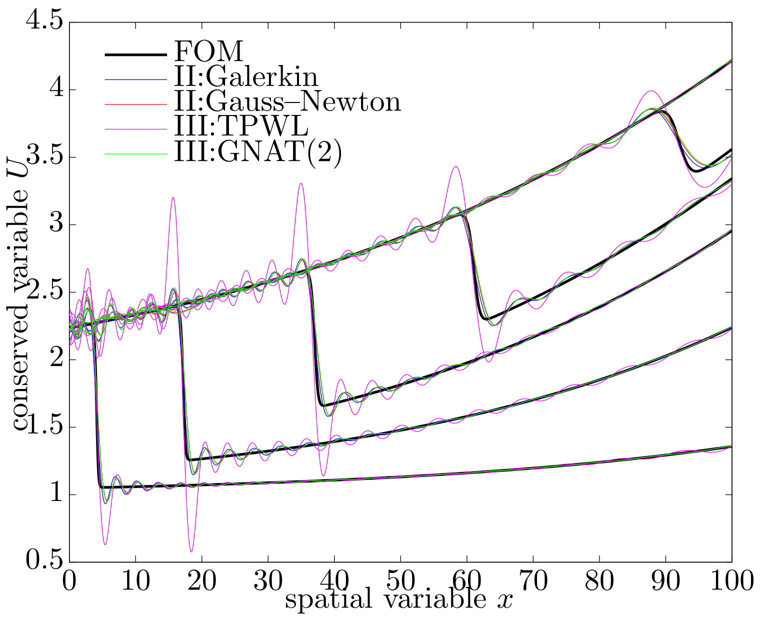


Figure 3. Performance of model reduction techniques on the one-dimensional Burgers' equation. Conserved variable plotted for $t = 2.5,\ 10,\ 20,\ 30,\ 40,\ 50$.

error (2.82%) is nearly as small as the Gauss–Newton solution, yet it generates a significant speedup of 59.7. This illustrates the ability of the GNAT to generate fast, accurate solutions for nonlinear fluid dynamics problems cast in the finite volume framework.

Figure 4 depicts the sample nodes for the sample mesh generated for this problem. Since Godunov's scheme is first-order in space and there is only one unknown per grid point, generating a sample mesh for this problem is tantamount to selecting grid points and then including the grid point's neighbors. For this problem, the selected nodes are concentrated near the inlet boundary condition. This can be attributed to the fact that the strength of the shock dissipates as it moves downstream (see Figure 4). Thus, more severe nonlinearities occur upstream, and the sample index selection algorithm identifies this region as requiring the most sampling.

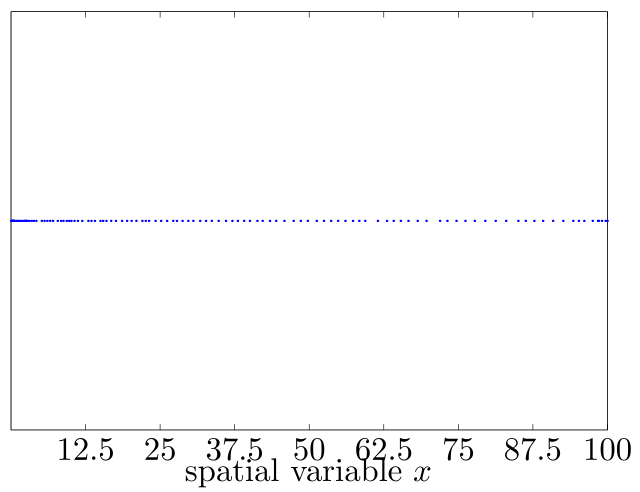


Figure 4. Sample nodes selected for the sample mesh for the one-dimensional Burgers' equation.

V.A.2. Prediction

This study tests the predictive ability of GNAT on the one-dimensional Burgers' equation. The input parameters are chosen to be the inlet boundary condition parameter a in Eq. (20) and the coefficient b in the source term of Eq. (18). Figure 5 depicts the training and online inputs used for this study. Since the GNAT model uses snapshot procedure 2, both the tier I and tier II models are run at all training inputs to generate the appropriate snapshots. This leads to a "global" GNAT model constructed from snapshots generated at three different training inputs.

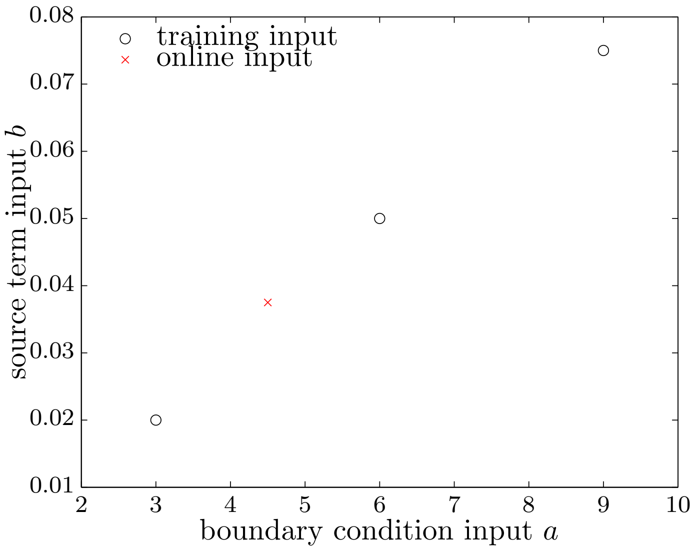


Figure 5. Offline and online inputs for GNAT applied to Burgers' equation

Figure 6 displays the results for this predictive problem; here, the full-order response at the online input is shown for comparison purposes. Note that the GNAT prediction closely matches the full-order response, producing an relative error of only 1.43% in the time-averaged Euclidean norm of the state vector. The speedup for the GNAT simulation compared with the full-order simulation is 9.4.

Note that although oscillations in the GNAT response are apparent at $t = 2.5$, they dissipate over time; this occurs in part because the POD subspace restricts the evolution of the state in such a way that these oscillations do not grow. This example illustrates the predictive capability of the GNAT method, as it generates a fast, accurate responses at a previously unqueried point in the input parameter space.

V.B. Ahmed body problem

To demonstrate the performance of the GNAT model reduction method on a large-scale, nonlinear fluid dynamics problem, consider as an example the airflow around the Ahmed body shown in Figure 7. This problem, which is a benchmark for turbulent flows around automobiles, was initially investigated by Ahmed *et. al*[30] in the 1980s. The mesh, shown in Figure 8, consists of 2,890,434 nodes and 17,017,090 tetrahedral volumes, resulting in $N = 17,342,604$ degrees of freedom. A symmetry plane is employed to exploit the symmetry of the body about the x–z plane. For all experiments, the slant angle is fixed at $\varphi = 20\,\text{deg}$. All numerical results presented in this section are not predictive—the online problem is identical to the training problem. As will be shown, re-creating the training results is not trivial for the large-scale, highly nonlinear, compressible, turbulent flow considered herein; future work will investigate predictive scenarios.

The full-order model corresponds to an unsteady Navier-Stokes simulation with DES turbulence model, Reichardt's wall law, V4 dissipation scheme, free-stream velocity $V_\infty = 60$ m/s, and Reynolds number $Re = \frac{\rho_\infty V_\infty}{\mu_\infty} = 4.29 \times 10^6$. The simulation employs a second-order accurate linear flux reconstruction and the second-order accurate implicit three-point backward difference scheme for time integration. A uniform step size of 8×10^{-5} seconds is taken, corresponding to a maximum CFL number of roughly 2000. Newton's method is employed to solve the nonlinear system of equations arising at each time step, and convergence

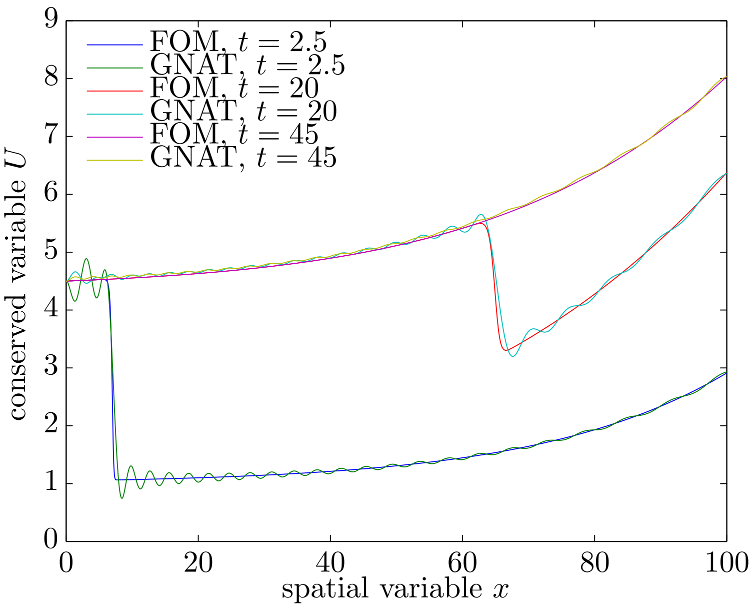


Figure 6. Predictive results for GNAT applied to Burgers' equation

is declared when the residual at the k^{th} iteration satisfies $R^{(k)} \leq 0.001R^{(0)}$. A steady-state simulation computes the initial condition. This steady-state computation is characterized by the same parameters as above, except that it employs local time-stepping with a maximum CFL number of 200, the first-order implicit backward Euler time integration scheme, and only one Newton iteration per (pseudo) time step. For this problem, the output of interest is the drag coefficient $C_D = \frac{D}{\frac{1}{2}\rho_\infty V_\infty^2 5.6016\times10^{-2}m^2}$ around the body.

All simulations are executed using AERO-F, which is a domain-decomposition-based, parallel, three-dimensional, compressible, Euler/Navier-Stokes solver developed by Farhat and co-workers. Simulation times are reported for a Linux cluster containing compute nodes with 16 GB of memory. Each node consists of two quad-core Intel Xeon E5345 processors running at 2.33 GHz inside a DELL Poweredge 1950. The cluster's interconnect is Cisco DDR InfiniBand. All simulation times are reported for the solution of the governing equations and the output of the state vector (full-order model) or generalized state coordinates (reduced-order models); a separate post-processing step computes outputs.

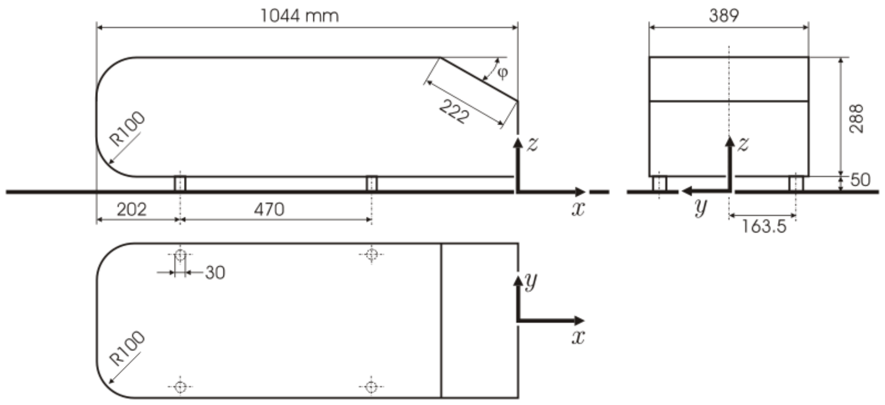
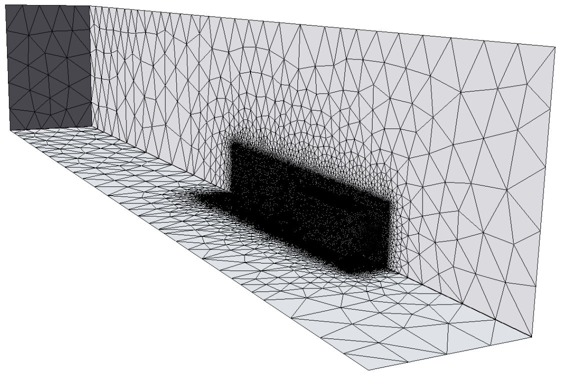


Figure 7. Ahmed body geometry (from Ref. 31).


(a) Entire computational mesh

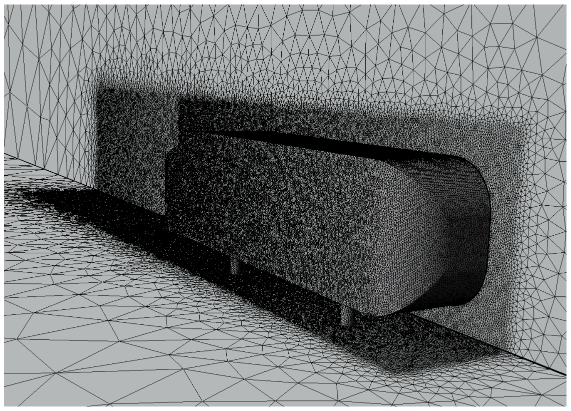

(b) Computational mesh near the body

Figure 8. Ahmed body mesh for $\varphi = 20\,\mathrm{deg}$ with 2,890,434 nodes and 17,017,090 tetrahedral volumes.

V.B.1. Comparison with experiment

Ref. 30 reports an experimental drag coefficient of 0.250 around the Ahmed body for a slant angle of $\varphi = 20\,\mathrm{deg}$. Figure 9 shows the time evolution of the drag coefficient generated by the full-order computational model for the same configuration. The time-averaged drag coefficient computed using the trapezoidal rule is 0.2524. This corresponds to a relative error of 0.95% with respect to the experimental results. As this error is below 1%, we declare this computational model to match the experimental results for the purpose of computing the drag coefficient.

This full-order model response is generated in 4.78×10^4 seconds (approximately 13.3 hours) using 512 cores, for a total cost in computational resources of 6,798 core-hours.

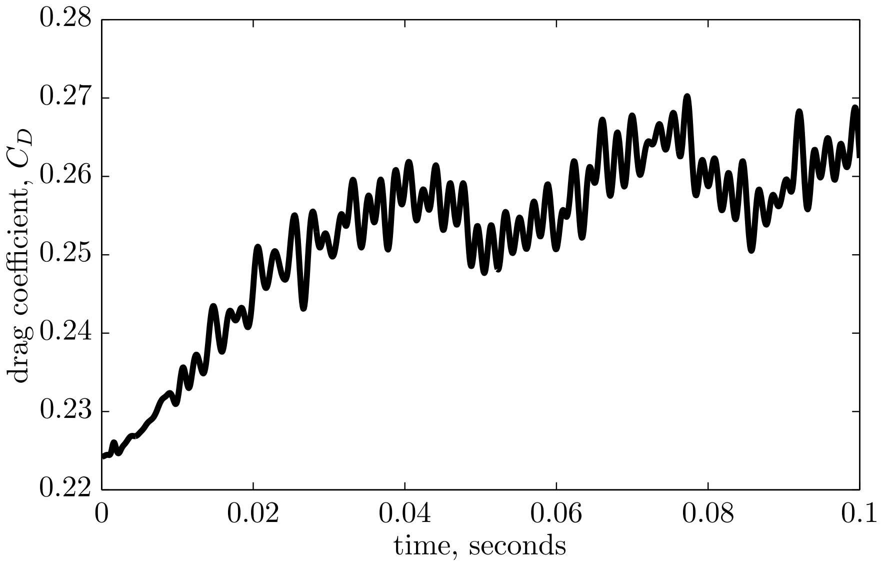


Figure 9. Drag coefficient generated by full-order model.

V.B.2. Snapshot method study

First, the GNAT method is analyzed for different residual/Jacobian snapshot procedures. Recall from Section III.D.2 that snapshot procedure 0 is inconsistent, but is similar to the approach most often taken in the literature. Procedure 1 satisfies one consistency condition.

To build the POD basis for the state, consistent snapshots $\{y^n - y^0\}_{n=1}^{n_t}$ with $n_t = 1252$ are collected during full-order simulation. Then, these snapshots are normalized such that each snapshot has unit magnitude and a POD basis is computed via the singular value decomposition. The number of POD basis vectors for the state is set to $n_y = 283$, which corresponds to 99.99% of the total statistical energy of the snapshots as measured by the sum of the squares of the singular values. All numerical studies in this section use this POD basis for the state.

To compute the drag around the Ahmed body for the GNAT simulations, the post-processing surface mesh shown in Figure 11 is used. It is characterized by 124,047 nodes (4.3% nodes of original mesh) and 492,445 tetrahedral volumes (2.9% volumes of original mesh). For all simulations, the post-processing output computation step (see Section IV.C for details) took 12.2 minutes using 4 cores, or 9.7 minutes using 8 cores.

The sample meshes generated for snapshot procedures 0 and 1 using Algorithm 2 with $n_{\mathsf{greed}} = 219$ and $n_{\mathsf{sample}} = 378$ are depicted in Figure 10. Both meshes are characterized by only 378 sample nodes; thus, the residual and Jacobian are computed at only 0.013% of the nodes in the original mesh. First, note that the algorithm selects one sample node from the inlet boundary face; this is designed into the algorithm to enable the GNAT model to enforce boundary conditions for the problem. Next, notice that the algorithm chooses sample nodes from three regions of the domain: the wake region behind the body, and the regions behind the two cylidrical experimental supports. This means that, on average, the magnitude of the residual is highest in these regions during the training simulations. This is consistent with the fact that the fluid flow in these parts of the domain is highly nonlinear and is characterized by separated flow with strong vorticity.

The GNAT models employ $n_J = n_R = 1514$, which corresponds to 99.99% of the energy in the snapshots of the residual collected during the tier II Gauss–Newton simulation. As with the snapshots of the state, the residual snapshots are normalized. Both GNAT simulations are executed using only 4 cores (compared with 512 cores used for the full-order model). This is possible because the sample mesh enables a very small subset of the computational domain to be loaded online.

Figure 12 and Table 3 contain the results for these simulations. Here (and in all remaining sections), the relative error is measured as:

$$\text{relative error} = \frac{\frac{1}{n_t}\sum_{n=1}^{n_t} |C_D^{\mathrm{I}}(n) - C_D^{\mathrm{ROM}}(n)|}{\max_n C_D^{\mathrm{I}}(n) - \min_n C_D^{\mathrm{I}}(n)}, \tag{21}$$

where $C_D^{\mathrm{I}}(n)$ is the drag coefficient at time step n computed by the full-order (tier I) model, and C_D^{ROM} is the drag coefficient computed by a reduced-order model. Note that both snapshot procedures 0 and 1 lead to GNAT models characterized by speedups (as measured in total computational resources) of at least 231. This occurs largely due to the drastic reduction in cores made possible by the sample mesh.

Next, notice that snapshot procedure 1 produces a GNAT model with a nearly exact response; snapshot procedure 0 leads to a significantly less accurate solution. Furthermore, snapshot procedure 1 leads to a much faster simulation time, as it requires many fewer Newton iterations for convergence at each time step. This illustrates the importance of consistency when constructing a reduced-order model.

snapshot procedure	relative error	average Newton iterations per time step	time, hours	speedup in computational resources
0	7.43%	6.47	7.37	231
1	0.68%	2.75	3.88	438

Table 3. Performance of GNAT using different snapshot procedures for 378 sample nodes. Simulation times reported using 4 cores.

V.B.3. Sample node study

This study analyzes the effect of the number of sample nodes on GNAT performance. Three sample meshes are generated using Algorithm 2 with $n_{\mathsf{greed}} = 219$ for 253 sample nodes, 378 sample nodes, and 505 sample nodes. Figure 13 displays these sample meshes and Table 4 reports their attributes. Note that all sample meshes employ less than 0.7% of the nodes from the original mesh and less than 0.4% of the tetrahedral volumes from the original mesh. The GNAT models for these simulations employ the same POD bases with $n_J = n_R = 1514$. Since there are 6 unknowns per node (five conserved state variables plus one turbulence model variable), the mesh with 253 sample nodes corresponds (roughly) to interpolation of the nonlinear functions ($253 \times 6 = 1518$ rows and 1514 columns in the $\widehat{\Phi_R}$ and $\widehat{\Phi_J}$ matrices). 378 sample nodes gives a least-squares "aspect ratio" of 1.5, where the aspect ratio is defined by the number of rows divided by the number of columns in the $\widehat{\Phi_R}$ and $\widehat{\Phi_J}$. Finally, 505 sample nodes leads to a value of 2.0 for this aspect ratio.

# sample nodes	# nodes	# primal volumes	fraction of nodes from full mesh	fraction of volumes from full mesh
253	12,808	41,014	0.44%	0.24%
378	17,096	56,280	0.59%	0.33%
505	19,822	67,082	0.69%	0.39%

Table 4. Sample mesh attributes for different numbers of sample nodes

Figure 12 and Table 5 contain the results for the GNAT model using these sample meshes. All sample meshes lead to a relative error less than 1%. Also, as sample nodes are added, the average number of Newton iterations decreases, indicating that adding sample nodes improves convergence. However, the fastest wall time performance is achieved for the smallest number of sample nodes (253); adding sample nodes increases simulation time for this problem because, even though it leads to fewer Newton steps, the residual and Jacobian are computed at more nodes.

These results imply that interpolation of the nonlinear functions (as is prescribed by the approaches taken in Refs. [18, 19, 25, 26, 29]) is not always the most computationally efficient approach. Rather, adding sample nodes can improve convergence and accuracy.

# sample nodes	relative error	average Newton iterations per time step	time, hours	speedup in computational resources
253	0.79%	4.38	3.77	452
378	0.68%	2.75	3.88	438
505	0.75%	2.25	4.22	403

Table 5. Performance of GNAT with snapshot procedure 1 for different numbers of sample nodes.

V.B.4. Parallel performance study

This section tests the parallel performance of the GNAT method. Table 6 presents timing results for the GNAT method using snapshot procedure 1 with 378 sample nodes and a different number of computing cores. All tests employ the same number of subdomains in the sample mesh partition (as computed by METIS) as cores; the exception is the 1-core case, which employs 2 subdomains.

# cores	time, hours	computational resources, core-hours	speedup in simulation time	speedup in computational resources
1	16.1	16.1	0.83	422
2	8.74	17.5	1.52	388
4	3.88	15.5	3.43	438
8	2.50	20.0	5.32	340
12	1.94	23.3	6.86	292
16	2.08	33.3	6.39	204

Table 6. Simulation times for GNAT with snapshot procedure 1 using 378 sample nodes for different numbers of cores.

These results indicate that the fastest wall time is obtained for 12 cores. This leads to a speedup as measured in wall time of 6.86. The simulation with 4 cores has the greatest speedup as measured in total computational resources; this value is 438.

This study demonstrates that the GNAT method is capable of generating very high speedups as measured in total computational resources for large-scale problems. Speedups in pure wall time—while non-negligible—are more modest. This occurs because the GNAT model corresponds to a "small" problem that saturates parallelism for a relatively small number of cores (12 in this case); when cores are added beyond this point, the additional overhead overrides added computing power.

V.B.5. Comparison with other function sampling methods

This study compares the performance of various complexity reduction techniques. All function sampling techniques are tested using the same POD basis for the state and the same sample mesh. The sample mesh uses 378 nodes and is depicted in Figure 10(b). The methods that are tested are:

1. GNAT with snapshot procedure 1 and $n_R = n_J = 1514$.
2. Collocation of the nonlinear equations followed by a Galerkin projection of the resulting overdetermined system of 2268 nonlinear equations (368 sample nodes $\times$ 6 equations per node) in $n_y = 283$ unknowns. Ref. 22 introduced this method.
3. Collocation followed by a least-squares solution of the overdetermined system. Ref. 21 proposed this approach.
4. A discrete empirical interpolation (DEIM)-like approach. Ref. 25 presented the DEIM approach. The specific approach tested here employs snapshot procedure 0 (residual snapshots collected during the full-order simulation) and $n_R = n_J = 2268$, which leads to interpolation of the residual and Jacobian. The tested approach employs the Gauss–Newton solution of the overdetermined equations as opposed to the Galerkin projection; this is done to isolate the effect of the complexity reduction technique on performance.

Figure 15 contains the drag coefficient responses generated using all complexity reduction methods. Both collocation approaches generate negative pressures after only a few time steps; at this point, the simulation stops. This demonstrates that collocation is insufficient for characterizing the behavior of the full nonlinear system, as it does not approximate the nonlinear functions in the entire domain.

The DEIM-like approach also does not perform well, as the Newton iterations begin to generate zero search directions after only a few time steps. This is likely due to two factors. First, the snapshots employed by DEIM are not consistent. As was shown in Section V.B.2, using inconsistent snapshots for the residual and Jacobian can lead to poor results. Secondly, the method employs interpolation as opposed to a least-squares fit of the nonlinear functions. Section V.B.3 showed that this can hamper convergence; in this case, the effect is more dramatic, as the method is likely "overfitting" the nonlinear functions using low-energy POD basis functions.[6]

VI. Conclusions

This work presented several developments in the GNAT nonlinear model reduction method. In particular, the sample mesh concept was introduced, as well as a post-processing output computation step. As was demonstrated in the Ahmed body example, both of these developments enabled the GNAT model to use many fewer computational cores compared with the full-order model. This is crucial to the method's performance on large-scale problems.

Results on the one-dimensional inviscid Burgers' equation highlighted GNAT's favorable performance compared with the TPWL method. The predictive ability of GNAT was also demonstrated, as a speedup of nearly 10 was achieved with a relative error less than 2% at a non-training point in the input space.

Numerical experiments for the compressible, turbulent flow around the Ahmed body on a model with millions of unknowns were also executed. First, these results showed that consistency seems to be important

[6]Recall that the DEIM-like approach employs 2268 basis vectors as opposed to GNAT, which uses 1514 basis vectors.

when constructing reduced-order models: snapshot method 0 (inconsistent) generated 10 times greater error than snapshot method 1 (consistent). Second, it was shown that least-squares reconstruction of the nonlinear functions can perform better than interpolation. This was apparent, as adding sample nodes (with other parameters held fixed) decreased the average number of Newton steps and led to improvements in error compared with the interpolatory approach. Finally, the GNAT method performed favorably compared with collocation and discrete empirical interpolation-like approaches; for a fixed sample mesh, GNAT was the only method that generated accurate results.

Future work includes pursuing predictive scenarios for large-scale problems, devising a method for choosing GNAT parameters (e.g. number of basis vectors) *a priori*, and investigating methods for further decreasing the wall time (not just computational resources) required for GNAT simulations.

Acknowledgments

The authors acknowledge partial support by the Motor Sports Division of the Toyota Motor Corporation under Agreement Number 48737, and partial support by the Army Research Laboratory through the Army High Performance Computing Research Center under Cooperative Agreement W911NF-07-2-0027. The first author also acknowledges partial support by the National Science Foundation Graduate Fellowship, and partial support by the National Defense Science and Engineering Graduate Fellowship. The content of this publication does not necessarily reflect the position or policy of any of these supporters, and no official endorsement should be inferred. The authors also thank Phil Avery for his contributions to the implementation of the model reduction method presented in this paper, and Wade Spurlock for his contributions to some of the simulations.

References

[1]Carlberg, K., Farhat, C., and Bou-Mosleh, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, April 2011, pp. 155–181.

[2]Antoulas, A., *Approximation of large-scale dynamical systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005.

[3]Prud'homme, C., Rovas, D., Veroy, K., Machiels, L., Maday, Y., Patera, A., and Turinici, G., "Reliable real-time solution of parameterized partial differential equations: Reduced-basis output bound methods," *Journal of Fluids Engineering*, Vol. 124, No. 1, 2002, pp. 70–80.

[4]Rozza, G., Huynh, D., and Patera, A., "Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations," *Archives of Computational Methods in Engineering*, Vol. 15, No. 3, 2007, pp. 1–47.

[5]Veroy, K., Prud'homme, C., Rovas, D., and Patera, A., "A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations," *AIAA Paper 2003-3847, 16th AIAA Computational Fluid Dynamics Conference, Orlando, FL*, June 23–26, 2003.

[6]Nguyen, N., Veroy, K., and Patera, A., *Certified real-time solution of parametrized partial differential equations*, Kluwer Academic Publishing, Dordrecht, 2005, pp. 1529–1564.

[7]Veroy, K. and Patera, A., "Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: Rigorous reduced-basis a posteriori error bounds," *International Journal for Numerical Methods in Fluids*, Vol. 47, No. 8, 2005, pp. 773–788.

[8]Hall, K. C., Thomas, J. P., and Dowell, E. H., "Reduced-order modelling of unsteady small-disturbance flows using a frequency domain proper orthogonal decomposition technique," *AIAA Paper 99-16520*, 1999.

[9]LeGresley, P. A. and Alonso, J. J., "Airfoil design optimization using reduced order models based on proper orthogonal decomposition," *AIAA Paper 2000-25450, Fluids 2000 Conference and Exhibit, Denver, CO*, June 19–22, 2000.

[10]Hall, K. C., Thomas, J. P., and Dowell, E. H., "Proper orthogonal decomposition technique for transonic unsteady aerodynamic flows," *AIAA Journal*, Vol. 38, No. 2, 2000, pp. 1853–1862.

[11]Willcox, K. and Peraire, J., "Balanced model reduction via the proper orthogonal decomposition," *AIAA Journal*, Vol. 40, No. 11, 2002, pp. 2323–2330.

[12]Epureanu, B. I., "A parametric analysis of reduced order models of viscous flows in turbomachinery," *Journal of Fluids and Structures*, Vol. 17, 2003, pp. 971–982.

[13]Thomas, J. P., Dowell, E. H., and Hall, K. C., "Three-dimensional transonic aeroelasticity using proper orthogonal decomposition-based reduced order models," *Journal of Aircraft*, Vol. 40, No. 3, 2003, pp. 544–551.

[14]Kim, T., Hong, M., Bhatia, K. B., and SenGupta, G., "Aeroelastic model reduction for affordable computational fluid dynamics-based flutter analysis," *AIAA Journal*, Vol. 43, No. 12, 2005, pp. 2487–2495.

[15]Lieu, T., Farhat, C., and Lesoinne, M., "Reduced-order fluid/structure modeling of a complete aircraft configuration," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, 2006, pp. 5730–5742.

[16]Lieu, T. and Farhat, C., "Adaptation of aeroelastic reduced-order models and application to an F-16 configuration," *AIAA Journal*, Vol. 45, 2007, pp. 1244–1269.

[17]Amsallem, D. and Farhat, C., "An Interpolation Method for Adapting Reduced-Order Models and Application to Aeroelasticity," *AIAA Journal*, Vol. 46, No. 7, July 2008, pp. 1803–1813.

[18]Grepl, M., Maday, Y., Nguyen, N., and Patera, A., "Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations," *M2AN (Math. Model. Numer. Anal.)*, Vol. 41, No. 3, 2007, pp. 575–605.

[19]Nguyen, N. and Peraire, J., "An efficient reduced-order modeling approach for non-linear parametrized partial differential equations," *International Journal for Numerical Methods in Engineering*, Vol. 76, February 2008, pp. 27–55.

[20]Rewienski, M., *A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems*, Ph.D. thesis, Citeseer, 2003.

[21]LeGresley, P., *Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods*, Ph.D. thesis, Stanford University, 2006.

[22]Astrid, P., *Reduction of Process Simulation Models: A Proper Orthogonal Decomposition Approach*, Ph.D. thesis, Technische Universiteit Eindhoven, 2004.

[23]Astrid, P., Weiland, S., Willcox, K., and Backx, T., "Missing point estimation in models described by proper orthogonal decomposition," *IEEE Transactions on Automatic Control*, Vol. 53, No. 10, 2008, pp. 2237–2251.

[24]Bos, R., Bombois, X., and Van den Hof, P., "Accelerating large-scale non-linear models for monitoring and control using spatial and temporal correlations," *Proceedings of the American Control Conference*, Vol. 4, 2004, pp. 3705–3710.

[25]Chaturantabut, S. and Sorensen, D., "Discrete Empirical Interpolation for Nonlinear Model Reduction," Tech. Rep. TR09-05, Department of Computational and Applied Mathematics, Rice University, March 2009.

[26]Galbally, D., Fidkowski, K., Willcox, K., and Ghattas, O., "Non-linear model reduction for uncertainty quantification in large-scale inverse problems," *International Journal for Numerical Methods in Engineering*, published online September 2009.

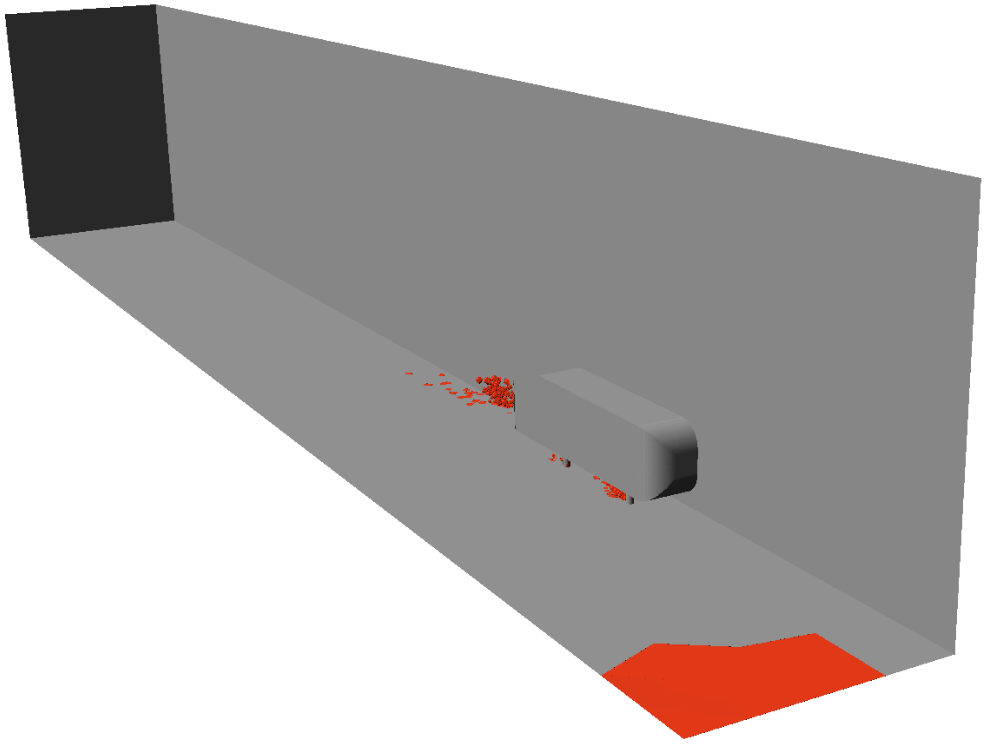
[27]Zahr, M., Carlberg, K., Amsallem, D., and Farhat, C., "Comparison of Model Reduction Techniques on High-Fidelity Linear and Nonlinear Electrical, Mechanical, and Biological Systems," Tech. rep., August 2010, Army High Performance Computing Research Center's Undergraduate Summer Institute in Computational Science and Engineering.

[28]Everson, R. and Sirovich, L., "Karhunen-Loeve procedure for gappy data," *Journal of the Optical Society of America A*, Vol. 12, No. 8, 1995, pp. 1657–1664.

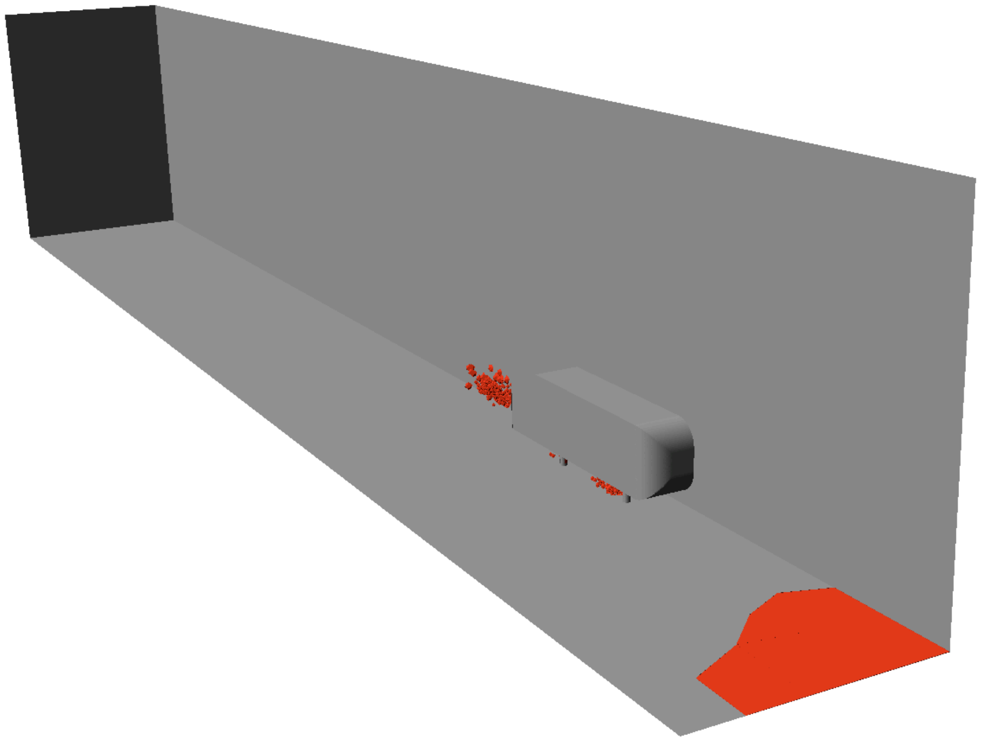
[29]Barrault, M., Maday, Y., Nguyen, N., and Patera, A., "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations," *Comptes Rendus Mathématique Académie des Sciences*, Vol. 339, No. 9, 2004, pp. 667–672.

[30]Ahmed, S., Ramm, G., and Faitin, G., "Some Salient Features of the Time-Averaged Ground Vehicle Wake," *SAE Paper 840300*, 1984.

[31]Hinterberger, C., Garcıa-Villalba, M., and Rodi, W., "Large eddy simulation of flow around the Ahmed body," *The Aerodynamics of Heavy Vehicles: Trucks, Buses, and Trains, Lecture Notes in Applied and Computational Mechanics*, edited by J. R. R. McCallen, F. Browand, Vol. 19, Springer, 2004.



(a) Sample mesh generated using snapshot procedure 0



(b) Sample mesh generated using snapshot procedure 1

Figure 10. Sample meshes generated using the specified snapshot collection procedure. Both meshes contain 378 sample nodes.

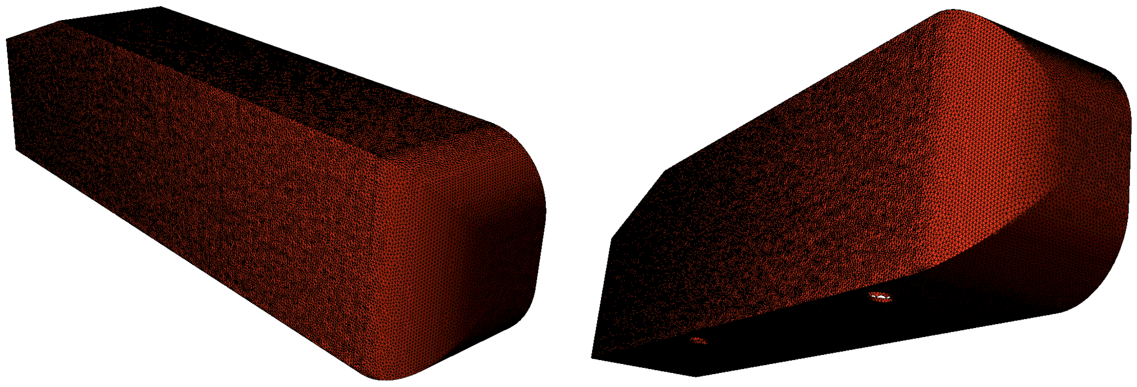


Figure 11. Surface mesh for post-processing with 124,047 nodes and 492,445 tetrahedral volumes.

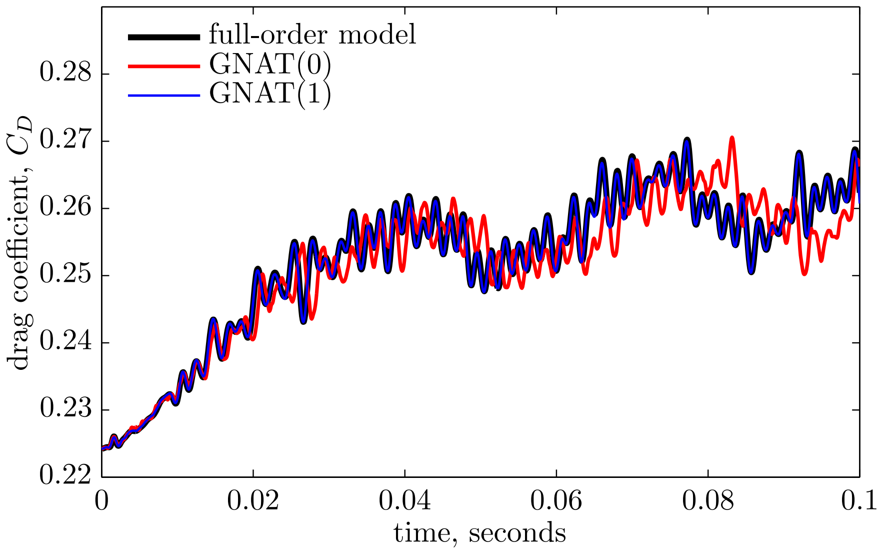
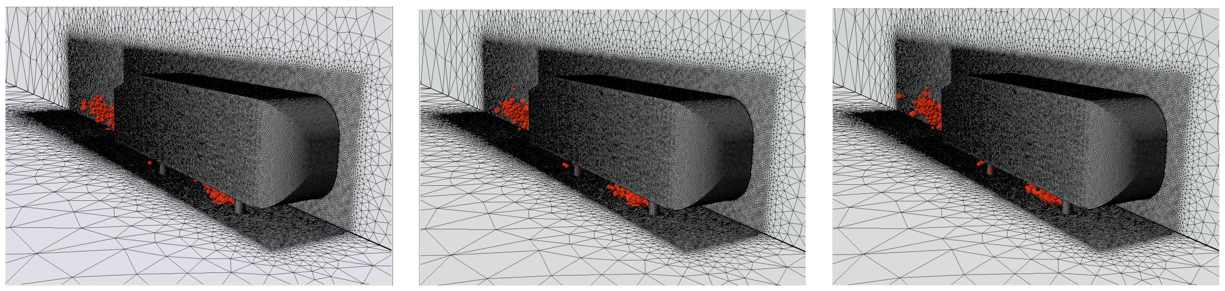


Figure 12. Drag coefficient generated by the GNAT model using 378 sample nodes and different snapshot procedures. GNAT(i) refers to GNAT with snapshot procedure i.



(a) 253 sample nodes (b) 378 sample nodes (c) 505 sample nodes

Figure 13. Sample meshes generated using snapshot method 1.

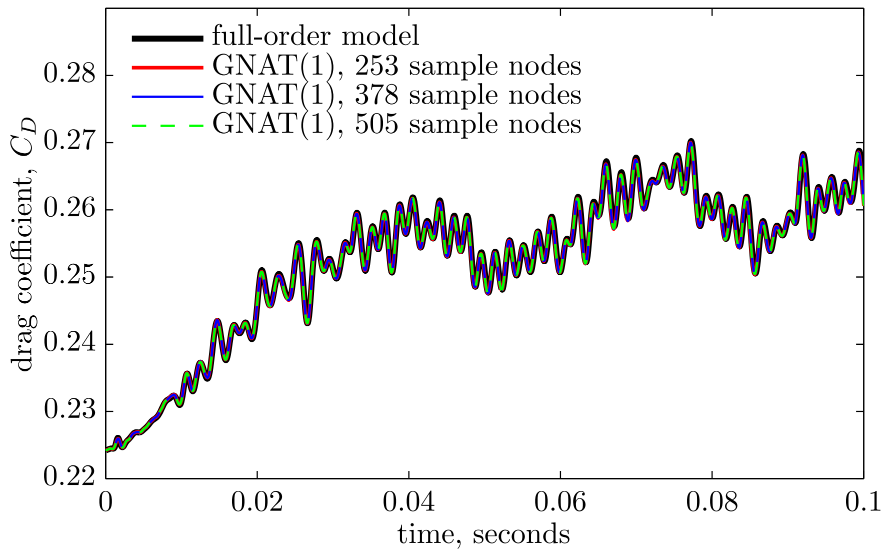


Figure 14. Drag coefficients generated by GNAT with snapshot procedure 1 for different numbers of sample nodes.

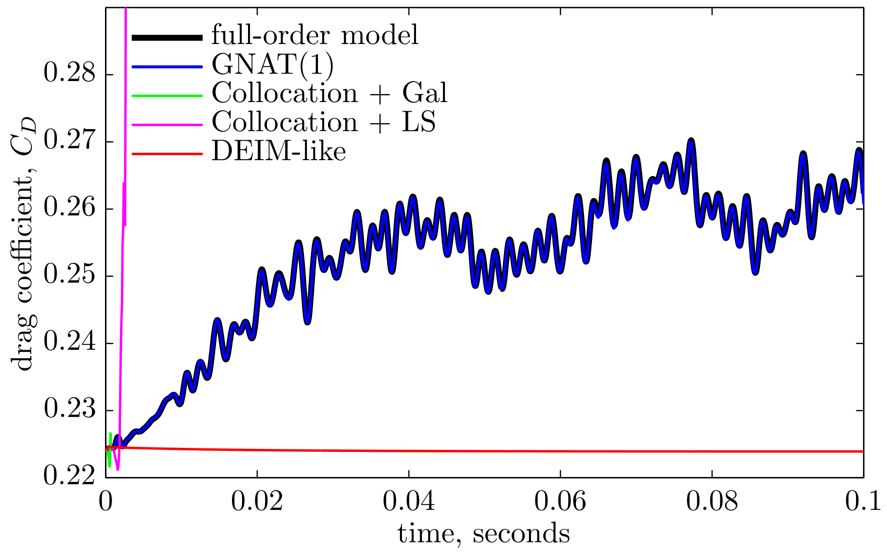


Figure 15. Drag coefficients generated by different complexity reduction techniques using the same POD basis and sample mesh with 378 sample nodes.